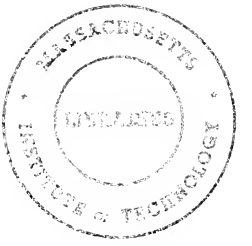
MASSACHUSETTS
INSTITUTE OF TECHNOLOGY
LIBRARIES

Stephen C. Graves

and

Bruce W. Lamar*

Alfred P. Sloan School of Management
Massachusetts Institute of Technology
Cambridge MA 02139

AN INTEGER PROGRAMMING PROCEDURE FOR ASSEMBLY SYSTEM DESIGN PROBLEMS

by

Stephen C. Graves

and

Bruce W. Lamar*

Alfred P. Sloan School of Management
Massachusetts Institute of Technology
Cambridge, MA 02139

December 1980

Revised June 1981

~ R&D 22: 12-24-81

* Presently at Arthur D. Little, Inc., Cambridge, Massachusetts.

ABSTRACT

Recent advances in robot technology have revolutionized the concept of manufacturing and assembly systems. These advances have created the need for new mathematical models to reflect the capabilities of the new technologies. In this paper, we focus on the system design problem by defining a work station selection and task assignment problem for automated assembly systems. The problem is formulated as a zero-one integer program and a procedure for seeking lower and upper bounds to the optimal value of the integer program is described. We show that the upper bound provides a feasible solution to the integer formulation and that the lower bound is tighter than the standard linear programming relaxation of the integer formulation. Computational results indicate that the proposed bounds are extremely tight. In fact, in each of the 42 test problems evaluated, the lower and upper bound coincided, indicating that the optimal solution to the integer program had been obtained.

With recent advances in manufacturing and computer technology, the notion of the automated factory has evolved from a science fiction fantasy to a reality. Some recent references in the popular press which describe these advances are Bylinsky (1979), Donlan (1980), Friedrich, et al. (1980), U.S. News and World Report (1979), Nicholson, et al. (1979), Business Week (1980), Schefter (1980). As these articles indicate, a wide variety of automated equipment for part fabrication, assembly, and material handling is now available for manufacturing and assembly systems. For instance, robots are being widely used in welding, spray-painting, and material-handling applications, while other types of automation such as numerically-controlled machining centers are performing part fabrication. In addition, robots are expected to work along side assembly workers in many future assembly systems. These technological advances have created a new set of opportunities for management scientists, with regard to both the design and the operational control of automated systems.

This paper focuses on the system design problem for automated assembly systems. An assembly system performs a set of distinct tasks for the assembly of a product family which typically consists of a primary product and its model variations. To perform these tasks, the system contains a set of work stations linked together by a transport mechanism for moving parts between stations. In addition, the assembly system includes the detailed specification of how a product's assembly flows from station to station. With these elements of an assembly system in mind, we see that decisions concerning the design of an assembly system may be divided into the following three components:

(a) The determination of the assembly requirements to be satisfied by the system. This includes specifying the product family to be assembled as well as identifying and sequencing the required assembly tasks.

(b) The selection of work stations and assignment of tasks to these stations. Here the designer not only determines the type and number of work stations, but also assigns the tasks to these stations.

(c) The selection of transport mechanisms and determination of the physical layout. The designer must find a feasible layout of the work stations, which includes the choice of a material handling system for transporting parts between stations and a routing plan for each part to be assembled.

This taxonomy of the system design process suggests a hierarchical approach in which the resolution of (a) is the primary input to (b), while the decisions from (b) constrain the choices in (c). Since the effects of the three decision levels are interrelated, the decision-making process would typically be implemented in an iterative fashion.

In this paper, we concentrate on (b), the work station selection and task assignment problem. The recent advances in automated assembly technology introduce new complexities and opportunities not addressed by the more traditional assembly design models such as the assembly line balancing problem (Ignall [1965]). In the line balancing problem we assume that all work stations are identical, and are to be laid out sequentially along a line. We then assign assembly tasks to these stations so as both to satisfy all precedence relationships between the tasks and to balance the total work load at the stations. In contrast, in the station selection and task assignment problem we select work stations from a set of nonidentical candidate work stations and simultaneously assign the assembly tasks to these selected stations to achieve a prespecified production rate at minimum cost. We do not presume a specific layout of these work stations so that considerable freedom to exchange product parts between stations in the assembly system is permitted. In addition, for automated assembly equipment, the capability for tool-changing is a distinguishing characteristic. Consequently we explicitly

include tool changing in the problem formulation. A more comprehensive treatment of our work may be found in Lamar (1980) which is an extension of the station selection and task assignment problem given in Graves and Whitney (1979). Future work will address parts (a) and (c) of the system design process.

The remainder of this paper is organized into five sections. The next section formulates the work station selection and task assignment problem as a zero-one integer program. We illustrate this formulation with an example based on the assembly of an automobile alternator. An approximate solution procedure for this integer program is presented in Sections 2 and 3. This solution procedure consists of solving two relaxations to the integer program to obtain lower bounds on the optimal value. These relaxations are also used to find near-optimal feasible solutions to the integer program and upper bounds on its optimal value. In Section 4 we report our computational experience that shows that the approximate procedure is extremely effective in finding the optimal solution to the integer program. Finally, in Section 5 we discuss possible model extensions and refinements.

1. PRESENTATION OF THE MODEL

In this section we formulate a zero-one integer linear program that selects work stations and assigns tasks to these work stations to minimize total cost in an assembly system which must satisfy a preset production rate. We then illustrate the model with the example of an automobile alternator assembly, and reformulate the model as a generalized programming problem.

Notation and Formulation

Let index set $I = \{1,\ldots,M\}$ denote the M possible work stations available for assembly operations and let index set $J = \{1,\ldots,N\}$ denote the N distinct tasks that must be performed by the assembly system. These N tasks may represent a single product or subassembly, or possibly a family of products or subassemblies. We define task 0 as a dummy "start-up" task and task N+1 as a dummy "completion" task. We assume that there is a strict ordering of the tasks such that task j must precede task k iff $j < k$. As a consequence, suppose that task 1, 2, and 4 are assigned to the first work station and task 3 is assigned to the second work station and must precede task 4. This implies that once the first work station performs tasks 1 and 2, the product must be unloaded and trasferred to the second work station for task 3 to be performed. Then the product is returned to the first work station to complete task 4.

Our formulation of the assembly system deisgn problem will select a subset from the set of work stations I and assign the tasks from set J to these work stations so as to minimize some cost criterion while achieving a preset production rate. This formulation requires four sets of parameters. First, for each work station i ($i \in I$), we specify an annualized fixed cost f_i representing the capital investment necessary to cover the work station and its required tooling. Second, we specify the amount of operating time b_i

available at work station i over an annual period. Typically, b_i would represent the expected "up-time" for the work station. Third, we define t_{ijk} ($i \in I$, $0 \leq j < k \leq N+1$) to be the amount of operating time necessary to satisfy the annual requirements for task j if work station i performs task j and task k is the next task performed at work station i. The parameter t_{ijk} typically consists of three components -- (i) the actual assembly time associated with task j, (ii) a load/unload time required for transferring the product to another work station if k does not equal j+1, and (iii) a tool change time if tasks j and k require distinct tools at work station i. By convention, t_{i0k} specifies the "startup" time at work station i when task k is the first task performed in each cycle at that work station, while $t_{i,j,N+1}$ gives the "completion" time when task j is the last task performed at work station i. Finally, similar to the definition of t_{ijk}, we define c_{ijk} ($i \in I$, $0 \leq j < k \leq N+1$) to be the annual operating cost for performing task j at work station i when task k is the next task performed at work station i. This cost includes the variable assembly cost, the variable handling cost, and the variable tooling cost.

If tasks $j,k,\ldots,\ell$ are assigned to work station i, the formulation, for purposes of calculating the annual operating time and cost at station i, assumes that the work schedule at station i is cyclic and follows the simple sequence $j,k,\ldots,\ell,j,k,\ldots$. To ensure the feasibility of this operating sequence, we assume that the system carries some minimal level of work-in-process inventory. In practice, it is possible to sequence the work at each station so as to minimize operating time or cost. Hence, our parameters should be thought of as conservative approximations to the actual costs and times.

We define two sets of decision variables as follows:

$$x_{ijk} = \begin{cases} 1 & \text{if task } j \text{ is performed at work station } i \text{ and task } k \text{ is the next task to be performed at work station } i \ (i\varepsilon I,\ 0 \leq j < k \leq N+1) \\ 0 & \text{otherwise.} \end{cases}$$

$$y_i = \begin{cases} 1 & \text{if work station } i \text{ is included in the assembly design configuration } (i\varepsilon I) \\ 0 & \text{otherwise.} \end{cases}$$

The integer programming formulation for the assembly system design problem is

$$\text{Problem P:} \quad \text{Min} \sum_{i=1}^{M} \left(f_i y_i + \sum_{j=0}^{N} \sum_{k=j+1}^{N+1} c_{ijk} x_{ijk} \right) \tag{1a}$$

subject to

$$\sum_{i=1}^{M} \sum_{k=j+1}^{N+1} x_{ijk} = 1 \qquad \forall j \varepsilon J \tag{1b}$$

$$\sum_{j=0}^{N} \sum_{k=j+1}^{N+1} t_{ijk} x_{ijk} \leq b_i \qquad \forall i \varepsilon I \tag{1c}$$

$$\sum_{k=j+1}^{N} x_{ijk} - \sum_{\ell=0}^{j-1} x_{i\ell j} = 0 \qquad \forall i \varepsilon I,\ j \varepsilon J \tag{1d}$$

$$\sum_{k=1}^{N+1} x_{i0k} - y_i = 0 \qquad \forall i \varepsilon I \tag{1e}$$

$$\sum_{j=0}^{N} x_{i,j,N+1} - y_i = 0 \qquad \forall i \varepsilon I \tag{1f}$$

$$x_{ijk} \varepsilon \{0,1\} \qquad \forall i \varepsilon I,\ 0 \leq j < k \leq N+1 \tag{1g}$$

$$y_i \varepsilon \{0,1\} \qquad \forall i \varepsilon I \tag{1h}$$

Problem P minimizes the total annual system cost given the annual production volume requirements. The objective function (1a) represents the annual capital and operating costs as a function of the decision variables. The first set of constraints (1b) specifies that each task j be assigned to a work station, while

constraints (1c) ensure that the amount of work assigned to a particular station does not exceed the time available at that station. Constraints (1d) through (1f) are conservation of flow constraints that enforce the precedence relationships of the tasks at each work station; thus, if task j is assigned to a work station i, then there exists an immediate predecessor task ℓ and successor task k to task j on station i (which may be the dummy tasks ℓ=0 or k=N+1). Furthermore, these constraints also relate the task assignment variables (x_{ijk}) with the work station selection variables (y_i); if y_i equals zero, so that station i is not included in the system design, then no task assignments are made to station i and the x_{ijk} variables are all zero.

To further highlight the nature of formulation P, we note that by representing the work stations as vehicles and the tasks as delivery points, we may interpret formulation P as a special type of vehicle routing problem (Dantzig and Ramser [1959]). Specifically, nonhomogenous vehicles are selected to make deliveries to nodes that are indexed such that they must be visited in increasing order of index. As described below, this precedence structure appears to make the problem easier to solve than the general vehicle routing problem and suggests a specialized solution procedure.

Example

In order to illustrate the use of formulation P, we apply the model to the assembly of an automobile alternator (Nevins and Whitney [1978]), which is depicted in the exploded view of Figure 1. The set of tasks that must be performed in order to assemble the alternator is described in Table 1. The precedence relationships of the tasks are prescribed by the task number; i.e., the tasks must be performed sequentially. Table 1 also specifies the tool requirements for each task. A set of five work stations, each being a computer-controlled robot, is available to perform the tasks listed in the

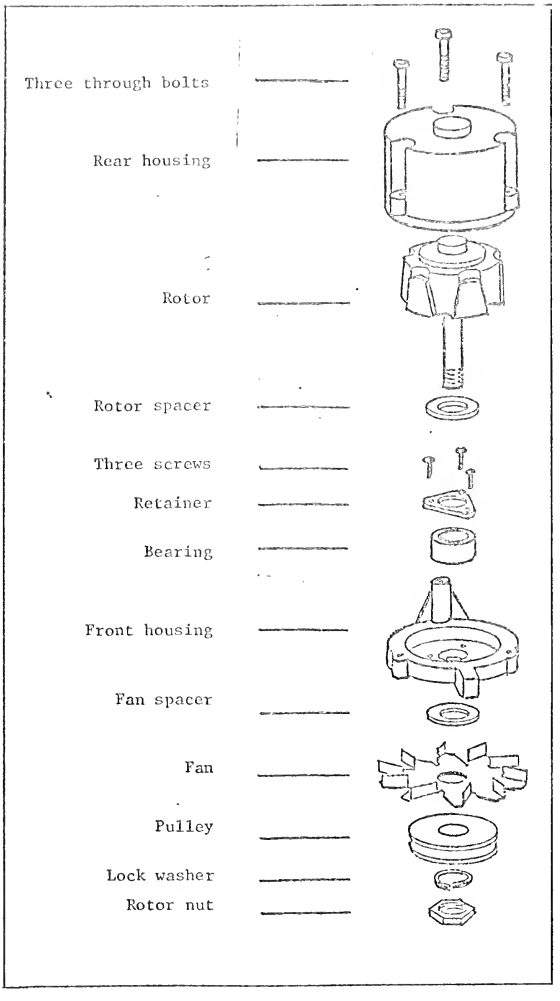

Figure 1: Exploded View of Automobile Alternator (Nevins and Whitney [1978])

Table 1: Description of Tasks and Tool Requirements for Automobile Alternator Example

Number of Task	Description of Task	Tool Required
1	Insert rotor nut	Small three-finger gripper
2	Insert lock washer	Small three-finger gripper
3	Insert pulley	Small three-finger gripper
4	Insert fan	Small three-finger gripper
5	Insert fan spacer	Small three-finger gripper
6	Insert front housing	Small three-finger gripper
7	Insert bearing	Small three-finger gripper
8	Insert retainer	Small three-finger gripper
9	Insert & tighten 3 screws	Special screwdriver
10	Insert rotor spacer	Small three-finger gripper
11	Insert rotor	Contracting collet
12	Tighten rotor nut	Rotor-nut tightener
13	Insert rear housing	Large three-finger gripper
14	Insert & tighten 3 through bolts	Bolt driver[(a)]

(a) Work station S_1 uses the special screwdriver to perform task 14.

table. These robots are of two types, general purpose and special purpose. The general purpose robots (referred to as G_1 and G_2) are able to perform most or all of the required tasks. They have the ability to use multiple tools and may be reprogrammed to meet changes in the production environment. They differ both in their capability to perform certain tasks and in their costs. In contrast, the three special purpose robots (referred to as S_1, S_2, and S_3), though less expensive than the general purpose robots, have limited flexibility and are capable of performing only a set of closely related tasks using a single tool. S_1 is a special screwdriver station, while S_2 and S_3 are pick-and-place robots that are specifically configured to do a subset of the tasks. See Graves and Whitney (1979) for a fuller description of this equipment.

We assume that each of the work stations has 120,000 minutes of available time annually, which are specified as the b_i parameters in the model. If a work station is selected, its time is consumed by performing assembly operations, loading and unloading the product, and changing tools. These three time components comprise the t_{ijk} parameter in formulation P. Naturally, an assembly time is incurred at a work station whenever that station performs a task. Since the precedence relationships of the tasks are sequential, a load/unload time is incurred at a work station only when that station performs the first task or when it performs a task j and does not perform task j+1. A tool-change time is incurred at a work station only when that station performs a task which requires a tool different from the tool required for the succeeding task performed at that work station.

Table 2 lists the annual assembly, load/unload, and tool-change times for the automobile alternator under consideration, assuming a production rate of one unit per minute for 120,000 minutes per year. We note that for robots G_1 and G_2 the tool changing time is comparable to task processing time. This

Table 2: Time Data for Automobile Alternator Example

Time Component	Work Station and Annual Time (1000's of minutes)[a]				
	G_1	G_2	S_1	S_2	S_3
Assembly Time[b]					
Task 1	12	8	-	6	-
Task 2	12	8	-	6	-
Task 3	12	8	-	6	-
Task 4	12	8	-	6	-
Task 5	12	8	-	6	-
Task 6	14	12	-	-	6
Task 7	12	8	-	-	6
Task 8	12	8	-	-	6
Task 9	8	8	6	-	-
Task 10	12	8	-	-	6
Task 11	12	-	-	-	-
Task 12	12	-	-	-	-
Task 13	12	-	-	-	-
Task 14	8	8	6	-	-
Load/Unload Time	2	2	2	2	2
Tool-Change Time[c]	12	8	-	-	-

(a) Based on a production rate of one unit per minute for 120,000 minutes per year.

(b) A dashed (-) entry indicates an infeasible task/work station combination.

(c) A dashed (-) entry indicates that a work station is unable to change tools.

is a realistic feature for these assembly robots and illustrates the importance of modeling tool changes in the system design problem. Although the timing data for the load and tool-change operations in the table are itemized only by work station, formulation P allows this data to be different for each work station and task.

In addition to timing information, formulation P requires cost information. This data is supplied in Table 3. The annual fixed costs correspond to the f_i parameters in P and represent the procurement of equipment, including necessary tooling. The annual variable costs correspond to the c_{ijk} parameters in P and reflect costs such as electricity, labor, and accessory costs which depend upon the volume of production. In Table 3 these costs again assume a production rate of one unit per minute for 120,000 minutes per year. Also, as with the timing data, although the variable costs are itemized only by the work station, this information may be made specific to both the work station and the task.

Using the data supplied above, the optimal solution to problem P is obtained using the techniques described in the following sections of this paper. The optimal total annual cost for this problem is \$106,300 and the optimal work station selections and task assignments are displayed in Table 4. According to this solution, the first five tasks should be performed on special purpose robot S_2. The alternator assembly should then be moved to special purpose robot S_3 which performs tasks 6 through 8. For task 9, the assembly should be moved to the general purpose robot G_1. Task 10 is performed by reloading the assembly onto robot S_3 and then the assembly moves back to robot G_1 for tasks 11 through 14 to be performed. The switching of the product between robots G_1 and S_3 for tasks 9, 10, and 11 occurs, in part, because G_1 does not have sufficient capacity to perform tasks 9 through 14. Table 4 also shows the percent of time that each of the selected

Table 3: Cost Data for Automobile Alternator Example

Cost Component	Work Station and Annual Cost ($)				
	G_1	G_2	S_1	S_2	S_3
Fixed Cost[(a)]	72000	46000	15000	15000	15000
Variable Cost[(b)]	500	500	100	200	200

(a) Based on a one-year payback period.

(b) Based on a production rate of one unit per minute for 120,000 minutes per year.

Table 4: Optimal Solution for Automobile Alternator Example

Work Station	Selected in Optimal Solution	Task Assignments in Optimal Solution	Work Station Utilization (percent)
G_1	yes	9, 11, 12, 13, 14	87
G_2	no	-	-
S_1	no	-	-
S_2	yes	1, 2, 3, 4, 5	27
S_3	yes	6, 7, 8, 10	23

work stations will be utilized during an annual period. Whereas G_1 is nearly fully utilized, both S_2 and S_3 are utilized less than 30%. Nevertheless, this is cheaper than any alternative solution.

We note that the above example is purely illustrative; indeed, it is quite simple and might be solved by inspection. In addition, we point out that P does not explicitly consider the physical layout of the work stations. Rather, in this model, we assume that a feasible layout of the work stations with appropriate part feeding devices and adequate buffer stocks may be constructed from the solution suggested by P. Accordingly, the solution to P is intended as a useful aid for the assembly system designer.

Reformulation of the Problem

Problem P is a very large pure integer program. Rather than attempting to solve P directly, we instead reformulate this problem as a generalized linear program (Magnanti, Shapiro, and Wagner [1976]) with binary variables. We then solve two relaxations of the reformulation - a pure linear relaxation and a mixed integer-linear relaxation. In the succeeding sections of this paper we show that these relaxations provide not only very tight lower bounds to P but also feasible candidate solutions to P that are near optimal.

To recast formulation P into a generalized programming framework, we consider constraints (1b) as the "complicating" constraints and the remaining constraints as the "easy" constraints. We observe that the "easy" constraints separate by index i into M subproblems. The feasible region for subproblem i is given by the set

$$X_i = \{(\underline{x}_i, y_i) : (\underline{x}_i, y_i) \text{ satisfies (1c) - (1h)}\} \quad \forall i \in I \qquad (2a)$$

where $\underline{x}_i$ denotes a vector with decision variable x_{ijk} as the $[1+j+k(k-1)/2]^{th}$ element of $\underline{x}_i$ for $0 \leq j < k \leq N+1$. Since each set X_i is bounded and constraints

(1g) and (1h) require that $\underline{x}_i$ and y_i be integer, it follows that each X_i contains a finite set of points. Hence, by letting R_i denote the cardinality of set X_i for $i \in I$, we may express X_i equivalently as

$$X_i = \{(\underline{x}_i^r, y_i^r) : r \in \{1,\ldots,R_i\}\} \qquad \forall i \in I \qquad (2b)$$

where $(\underline{x}_i^r, y_i^r)$ is the r^{th} point in set X_i. We let x_{ijk}^r denote the $[1+j+k(k-1)/2]^{th}$ element of vector $\underline{x}_i^r$ for $0 \leq j < k \leq N+1$.

To define the parameters for the reformulation, we observe that, if y_i^r equals zero, then work station i is not used and $\underline{x}_i^r$ is the zero vector; otherwise, for y_i^r equal to one, vector $\underline{x}_i^r$ may be interpreted as a feasible assignment of tasks to work station i. The cost associated with this assignment is given by

$$d_{ir} = \sum_{j=0}^{N} \sum_{k=j+1}^{N+1} c_{ijk} x_{ijk}^r \quad . \qquad (3)$$

We also define

$$a_{ijr} = \sum_{k=j+1}^{N+1} x_{ijk}^r$$

as a zero-one parameter to indicate whether or not task j is included in the r^{th} feasible assignment of tasks to work station i. With this notation, the generalized linear programming reformulation of problem P becomes

Problem Q: $$\text{Min} \sum_{i=1}^{M} (f_i y_i + \sum_{r=1}^{R_i} d_{ir} \lambda_{ir}) \qquad (4a)$$

subject to

$$\sum_{i=1}^{M} \sum_{r=1}^{R_i} a_{ijr} \lambda_{ir} = 1 \qquad \forall j \in J \qquad (4b)$$

$$\sum_{r=1}^{R_i} \lambda_{ir} - y_i = 0 \qquad \forall i \in I \qquad (4c)$$

$$\lambda_{ir} \in \{0,1\} \qquad \forall i \in I,\ r \in \{1,\ldots,R_i\} \qquad (4d)$$

$$y_i \in \{0,1\} \qquad \forall i \in I \qquad (4e)$$

where decision variable λ_{ir} is a zero-one variable denoting whether or not the r^{th} task assignment given by vector $\underset{\sim}{x}_i^r$ is performed at work station i.

Formulations P and Q are equivalent since there is a one-to-one correspondence between their feasible solutions, and, since, by (3), the objective function values for corresponding feasible solutions in the two formulations are equal. Moreover, given a feasible solution to Q, we may construct the corresponding feasible solution to P, and vice versa.

2. PURE LINEAR RELAXATION

Like problem P, problem Q is a very large pure integer formulation. Accordingly, in this section and the succeeding section, we consider two relaxations of problem Q. In this section, we describe the first relaxation, referred to as problem Q^1, in which we replace the integrality contraints (4d) and (4e), respectively, with

$$0 \leq \lambda_{ir} \leq 1 \qquad \forall i \in I,\ r \in \{1,\ldots,R_i\} \qquad (5a)$$

$$0 \leq y_i \leq 1 \qquad \forall i \in I \qquad (5b)$$

Our interest in problem Q^1 is threefold. First, by design, this problem is a pure linear program whose solution procedure fits naturally into a generalized programming algorithm (i.e. Dantzig-Wolfe decomposition). Second, the optimal value of Q^1 provides a tighter lower bound to the optimal value of P than does the linear programming relaxation of P. Third, Q^1 is used to seek a feasible solution to the pure integer problem P, and hence an upper bound to the optimal value of P.

Solution Procedure

To solve Q^1, we consider a column generation procedure (Dantzig and Wolfe [1961]), in which we require an efficient means of evaluating each of the M subproblems. For each subproblem i, the feasible region is given by the set X_i defined in (2); the objective function is

$$v_i(\underline{\pi}) = f_i y_i + \sum_{j=0}^{N} \sum_{k=j+1}^{N+1} (c_{ijk} - \pi_j) x_{ijk} \qquad (6)$$

where $\underline{\pi} = (\pi_j)$ is the vector of dual multipliers associated with the constraint set (4b), and $\pi_0 \equiv 0$. Thus, subproblem i for $i \in I$ is

Problem S_i: Min $v_i(\underline{\pi})$

subject to $(\underline{x}_i, y_i) \in X_i$

We observe that, for $y_i = 0$ we have $\underline{x}_i = \underline{0}$ and $v_i(\underline{\pi}) = 0$. When $y_i = 1$, S_i is an acyclic shortest path problem together with a single time availability constraint. One approach to solving problems of this structure has been suggested by Handler and Zang (1978), who maximize a Lagrangean relaxation of S_i in which the complicating constraint (1c) is relaxed; they resolve duality gaps by branch-and-bound. We consider an alternative approach which, for each index i, solves the subproblem with constraints (1c) through (1h) as an expanded acyclic shortest path problem and hence guarantees an integer solution.

To solve S_i for $y_i = 1$, we construct a network G_i with node set N_i and arc set A_i. Node set N_i contains ordered pairs of elements (j,ℓ) in which the first component represents a task assigned to station i and the second component is the cumulative time consumed at work station i. Symbolically,

$$N_j = \{(j,\ell) : j \in \{0,\ldots,N+1\},\ \ell \in \{0,\ldots,b_i\}\}. \tag{7}$$

The arcs contained in the arc set A_i are of the generic form $[(j,\ell),\ (k,m)]$, corresponding to the performance of task j at work station i at time ℓ followed by the performance of task k at work station i at time m. The strict precedence relationships of the tasks are maintained in network G_i by requiring that

$$k \begin{cases} > j & \text{if } j \in \{0,\ldots,N\} \\ = j & \text{if } j = N+1 \end{cases} \tag{8a}$$

The time consumed at work station i by performing task j when task k is the next task performed at that work station is represented in network G_i by letting

$$m = \begin{cases} \ell + t_{ijk} & \text{if } j \in \{0,\ldots,N\} \\ \ell + 1 & \text{if } j = N+1. \end{cases} \tag{8b}$$

Hence the arc $[(j,\ell),(k,m)] \in A_i$ if and only if $(j,\ell),(k,m) \in N_i$ and $(j,\ell),(k,m)$ satisfy (8). A unit flow on arc $[(j,\ell),(k,m)]$ corresponds to setting $x_{ijk} = 1$; thus, the arc-cost for arc $[(j,\ell),(k,m)]$ is $(c_{ijk} - \pi_j)$.

Using this structure, each path from node (0,0) to node $(N+1,b_i)$ in network G_i corresponds to a solution to S_i in which $y_i = 1$. Stated another way, each path represents an integral assignment of tasks to work station i which satisfies both the precedence relationships (1d) through (1f) and the time availability constraints (1c) at work station i. Hence, we solve S_i with $y_i = 1$ by finding the shortest path on G_i. The shortest path problem may be readily solved by simultaneously constructing and evaluating network G_i using a forward dynamic programming procedure. Assuming that the parameters b_i and t_{ijk} are integer, this procedure has a computational complexity of $O(N^2 \cdot b_i)$ operations (Lawler [1976]). If the optimal value of this shortest path problem is less than $-f_i$, then these values are the optimal solution to S_i; otherwise the optimal solution to S_i is $y_i = 0$ and $\underline{x}_i = \underline{0}$.

Lower Bound

We wish to show that problem Q^1 provides a tighter lower bound to the optimal value of the integer problem P than the linear programming relaxation of P. For notational convenience, we let $\bar{P}$ denote the linear programming relaxation of P and let v(A) denote the optimal value of any problem A. Using this notation, we claim that

$$v(\bar{P}) \leq v(Q^1) \leq v(P) \tag{9}$$

The right inequality of (9) is trivial to obtain since, as we have observed,

problems P and Q are equivalent and problem Q^1 is a relaxation of Q.

To demonstrate the validity of the left inequality of (9), we consider the Lagrangean relaxation (Shapiro [1979]) $L(\underline{\pi})$ of P by dualizing constraints (1b) to obtain

$$\text{Problem } L(\underline{\pi}): \quad \text{Min} \sum_{i=1}^{M} [f_i y_i + \sum_{j=0}^{N} \sum_{k=j+1}^{N+1} (c_{ijk} - \pi_j) x_{ijk}] + \sum_{j=1}^{N} \pi_j \qquad (10)$$

subject to (1c) through (1h),

where $\underline{\pi} = (\pi_j)$ is the vector of Lagrangean multipliers of constraint (1b) with $\pi_0 \equiv 0$. The optimal value of $\underline{\pi}$ is obtained by solving the dual problem

$$\text{Problem D:} \quad \max_{\underline{\pi}} v[L(\underline{\pi})].$$

As shown in Magnanti, Shapiro, and Wagner (1976), we may rewrite problem D as a linear program whose dual problem is given by Q^1. Thus, we have $v(D) = v(Q^1)$. In addition, we note that the linear programming relaxation of $L(\underline{\pi})$ is, in general, not equal to $L(\underline{\pi})$. This implies that Geoffrion's integrality property (1974) does not hold so that $v(\bar{P}) \leq v(D)$. Therefore, by simple substitution, we obtain $v(\bar{P}) \leq v(Q^1)$. Moreover, computational results show that $v(\bar{P}) < v(Q^1)$ is possible.

Upper Bound

The generalized programming procedure used to solve Q^1 may also be employed to seek an upper bound to the optimal value of problem P. Specifically, in each iteration of the generalized programming procedure, we test whether or not the decision variables $\{\lambda_{ir}\}$ are integer. If so, then the solution is a feasible solution to P and the corresponding objective function value is an upper bound to v(P).

If, in the process of solving Q^1, we do not find an integer solution at

any iteration, we may employ a heuristic procedure to obtain a solution feasible in P. Assume that the optimal solution to Q^1 contains a fractional task assignment; that is $0 < \lambda_{ir} < 1$ for at least one i and r. To resolve the fractional task assignments and construct a feasible solution to P, we find the largest fractional variable, say $\lambda_{\hat{i}\hat{r}}$, and round it up to one; we reset $\lambda_{\hat{i}r}$ to zero for all $r \neq \hat{r}$. This results in making the task assignment corresponding to vector $\underset{\sim}{x}^{\hat{r}}_{\hat{i}}$ to work station $\hat{i}$. We then select the next largest fractional variable and repeat the process until either all tasks are assigned to some work station or all available work stations have been selected. In the former case, if a task is assigned to more than one work station, then that task is reassigned solely to the work station with minimal operating cost; in this manner a feasible solution is easily found. In the latter case, this heuristic has failed to find a feasible solution; but this failure is not a serious worry since we can always find a feasible solution by expanding the candidate set I to permit more work stations.

If a feasible solution to P is obtained, then its objective function value may be compared to the lower bound for v(P) to obtain a conservative a posteriori measure of the near-optimality of the candidate solution. Computational experience has indicated that not only do these upper bound procedures nearly always generate a feasible solution to P, but also that the objective function values of these solutions are very close to the optimal value of P.

3. MIXED INTEGER-LINEAR RELAXATION

In the preceding section we demonstrated that problem Q^1, the linear programming relaxation of problem Q, may be used to seek lower and upper bounds to v(P), the optimal value of the integer program P. In this section, we consider a second relaxation, referred to as problem Q^2, in which we linearize the λ_{ir} variables but maintain the integrality requirements for the y_i variables used to select work stations. In other words, problem Q^2 is problem Q with constraints (4d) replaced with constraints (5a). This second formulation merits investigation since it provides a tighter lower bound to v(P) than does Q^1, and its solution permits us to generate additional candidate solutions to P. Of course, since Q^2 is a mixed integer-linear formulation, its solution requires a greater computational effort than that for Q^1. Below, we outline the solution procedure employed to solve Q^2 and describe the method of determining lower and upper bounds to v(P). The procedure for seeking the upper bound also supplies a feasible candidate solution to problem P.

Solution Procedure

Problem Q^2 has M integer variables, namely the y_i variables for $i \in I$. Thus, we may solve this relaxation by employing a standard branch-and-bound procedure (Garfinkel and Nemhauser [1972]) on these variables. The binary enumeration tree for the branch-and-bound procedure consists of M levels in which each level specifies a y_i variable as either zero or one. For notational convenience, we define the sets

$$I_0 = \{i : y_i = 0,\ i \in I\} ,$$

$$I_1 = \{i : y_i = 1,\ i \in I\} .$$

Using this notation, a node in the enumeration tree may be specified by the pair (I_0, I_1) where I_0, I_1 are disjoint sets. For each such node, we define problem Q^2 with constraints (4e) replaced with

$$y_i = 0 \qquad \forall i \in I_0 \qquad (11a)$$

$$y_i = 1 \qquad \forall i \in I_1 \qquad (11b)$$

$$y_i \in \{0,1\} \qquad \forall i \in I - (I_0 \cup I_1) \qquad (11c)$$

We let $\bar{Q}^2(I_0,I_1)$ denote the linear programming relaxation of $Q^2(I_0,I_1)$ in which constraints (11c) are linearized.

The branch-and-bound procedure enumerates, either explicitly or implicitly, all nodes (I_0,I_1) in which $I_0 \cup I_1 = I$. We initiate the procedure by solving Q^1 which is equivalent to solving $\bar{Q}^2(I_0,I_1)$ with $I_0 = I_1 = \emptyset$. We then evaluate problem $\bar{Q}^2(I_0,I_1)$ at each node (I_0,I_1) either by solving the linear program via the generalized programming procedure described in Section 3 or by solving its dual problem via a subgradient optimization of a Lagrangean relaxation (Shapiro [1979]). In the latter case, we evaluate the Lagrangean relaxation given by (10) in which the integer restriction on the y_i variables in (1h) are replaced with (11). Based on the value of $\bar{Q}^2(I_0,I_1)$, the branch-and-bound procedure may fathom a node (and all of its descendents) in the enumeration tree if $v[\bar{Q}^2(I_0,I_1)]$ exceeds the value of the best known candidate solution to Q^2; in addition, a node is also fathomed if the solution to $\bar{Q}^2(I_0,I_1)$ is feasible in Q^2, which yields a new candidate solution. If the procedure cannot fathom a node (I_0,I_1) by solving $\bar{Q}^2(I_0,I_1)$, then the procedure "branches" on a y_i variable in which $i \notin I_0 \cup I_1$ to form two new nodes - one in which y_i is one and the other in which y_i is zero. We note that when $I_0 \cup I_1 = I$, the solution to $\bar{Q}^2(I_0,I_1)$ always fathoms node (I_0,I_1) since $\bar{Q}^2(I_0,I_1) = Q^2(I_0,I_1)$ and so any solution to $\bar{Q}^2(I_0,I_1)$ is also feasible in Q^2. The branch-and-bound procedure continues until all nodes are fathomed, at which point the best candidate solution obtained is the optimal solution to Q^2.

In solving $\bar{Q}^2(I_0,I_1)$, the choice between using generalized linear

programming versus solving a dual problem via a subgradient optimization procedure depends on the desire for generating feasible candidate solutions to the integer program P. The subgradient optimization procedure is very efficient and provides a lower bound to $\bar{Q}^2(I_0,I_1)$ at each iteration; however, since it solves the dual problem, this procedure yields a feasible solution to $\bar{Q}^2(I_0,I_1)$ only when the optimal solution is found. On the other hand, the generalized linear programming procedure is a primal procedure, which is less efficient but which finds a feasible solution to $\bar{Q}^2(I_0,I_1)$ at each iteration. These intermediate solutions are also feasible in P if all the λ_{ir} variables are integer. In addition, the heuristic procedure described in Section 2 may also be used to seek a feasible solution to P. Accordingly, we use the primal procedure to evaluate $\bar{Q}^2(I_0,I_1)$ when $I_0 = I_1 = \emptyset$ or when $I_0 \cup I_1 = I$; at all other nodes, we use the subgradient evaluation method. We have found such a procedure to be very effective in finding near-optimal candidate solutions to P.

In summary, our solution procedure for P is a heuristic procedure based on the two relaxations, Q^1 and Q^2. We first solve Q^1 by a column generation procedure to obtain a lower bound on v(P). In this process we generate feasible candidate solutions to P, the best of which provides an upper bound on v(P). If this upper bound is sufficiently close to the lower bound, we stop the procedure. Otherwise, the second relaxation Q^2 is solved by a branch-and-bound procedure to obtain a better lower bound on v(P). Again, in this process, we seek feasible solutions to P, the best of which we take as our solution. The near-optimality of the best feasible solution can be determined by comparison with the lower bound given by $v(Q^2)$.

4. COMPUTATIONAL PERFORMANCE

We now examine the computational effectiveness of the solution procedure. To do this, we solved a series of test problems which used the alternator example, presented in Section 1, as a prototype. Below, we present the experimental plan undertaken to generate the test problems and report the computational effort required in their solution.

Experimental Plan

We consider a production setting in which three general purpose types of work station and five special purpose types of work station are available to assemble a product requiring the completion of up to 20 tasks. The Appendix contains the cost data, time data, and tool requirements for the test problems. This data is based on product information from the automobile alternator example, and on available cost and performance information for robots currently available to industry. Three different characteristics - the number of multiple general purpose robots, the number of tasks, and the production rate - are varied in the experimental plan in order to evaluate the effectiveness of relaxations Q^1 and Q^2 under alternative problem specifications. Changes in the number of work stations and tasks vary the complexity of the problems while changes in the production rate simulate alternative production environments.

The first characteristic is the number of general purpose work stations. It is often necessary to allow the selection of more than one work station of a particular type. One way to represent multiple work stations is to define a separate binary y_i variable for each of the identical work stations. Hence to permit the choice of up to seven identical work stations requires seven y_i variables. However, an alternative method for representing identical work stations is to define multiple work stations consisting of 2^n identical work stations, n=0,1,2,.... For instance, a double work station (n=1) repre-

sents two identical work stations, and has twice the fixed cost and twice the capacity of a single station. Hence to permit the choice of up to seven identical work stations requires the specification of only three multiple work stations: a single (n=0), a double (n=1), and a quadruple (n=2) work station. Thus, the second method permits representation of multiple work stations with fewer y_i variables.

In the experimental plan, the number of general purpose work stations is divided into three cases. The first case allows at most one of each general prupose work stations, so that M equals 8. The second case allows at most three of each general purpose work stations. This is done by defining a double station for each general purpose work station and thus M equals 11. The third case allows at most seven of each general purpose work station and is modelled in a similar manner so that M equals 14. The second characteristic in the experimental plan, the number of tasks, is divided into five categories by considering selected subsets of the 20 tasks. These five categories contain two categories in which N is 10 (tasks 1-10 and tasks 11-20), two categories in which N is 15 (tasks 1-15 and tasks 6-20), and one category with N equal to 20 (tasks 1-20). The third characteristic, the production rate, is set at three levels - 2.0, 4.0, and 6.0 units per minute.

In summary, implementation of this experimental plan results in the investigation of a total of 45 test problems ranging in size from 8 to 14 work stations and from 10 to 20 tasks.

Results

For each of the 45 test problems, we solve the two relaxations, Q^1 and Q^2, in order to examine both the tightness of the lower and upper bounds to the optimal value of the integer formulation P as well as the computational effort required to obtain these bounds. Of the 45 test problems, 42 are feasible in formulation P. Table 5 reports the deviation of the lower and

Table 5: Tightness of Lower and Upper Bounds for Test Problems

Relaxation	Absolute Deviation of Bounds from Optimal Value of Problem P (%) [a]							
	Lower Bound				Upper Bound			
	Maximum	Minimum	Average	Median	Maximum	Minimum	Average	Median
Q^1	20.0	0.0	4.1	0.0	15.5	0.0	1.5	0.0
Q^2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

(a) Based on 42 feasible test problems.

Table 6: Computational Performance for Test Problems

Relaxation	Computational Performance Measures [a]							
	Total Running Time (CPU seconds) [b]				Number of Problems Evaluated in Enumeration Tree [c]			
	Maximum	Minimum	Average	Median	Maximum	Minimum	Average	Median
Q^1	393.0	10.0	59.7	38.0 - 40.0	-	-	-	-
Q^2	1086.0	10.0	151.6	68.0 - 70.0	198.0	0.0	28.5	0.0

(a) Based on 42 feasible test problems.

(b) Using a PRIME 400 minicomputer.

(c) Excludes the initial linear program $\bar{Q}^2(\emptyset,\emptyset)$.

upper bounds obtained by solving Q^1 and Q^2 from v(P), the optimal value of P. For problem Q^1, although the lower bound ranges as much as 20 percent below v(P) and the upper bound ranges as much as 16 percent above v(P), these bounds are equal in over one-half of the feasible test problems. Even more interesting is the fact that in 33 of the 42 problems solved by Q^1, the candidate solution associated with the upper bound is the optimal solution to P (as verified by solving Q^2). Thus, in three-fourths of the problems, the optimal solution to the pure integer problem P is obtained simply by solving the pure linear problem Q^1.

For the test problems in which the lower and upper bounds obtained by solving Q^1 do not coincide, application of Q^2 yields a further refinement of these bounds. Remarkably, after completing the branch-and-bound procedure used to solve Q^2, the lower and upper bounds are equal for each of the 42 problems, and hence the optimal solution to P is obtained. The generalized programming procedure is by far the most common method of obtaining the upper bound. It produces the best upper bound in four-fifths of the feasible test problems while the heuristic procedure produces the best upper bound in only one-fifth of these problems.

Although solving Q^2 yielded the optimal solution to P for all test problems, examples exist where solving Q^2 does not solve P. In addition to this computational experiment, we have used this solution procedure for a few assembly system design projects for actual industrial products. For these projects we have occasionally (roughly once in every 15-20 problem instances) found that solving Q^2 did not solve P. However, in all instances the solution procedure generated a feasible solution to P with a solution value within 2% of $v(Q^2)$, which is a lower bound to v(P).

The computational effort required to solve each of the 42 feasible test problems by Q^1 and Q^2 is supplied in Table 6. The first measure of the computational effort provided in the table is the running time in cpu seconds on a

PRIME 400 time-shared minicomputer. Although cpu time is machine dependent (1 cpu second on a PRIME 400 is roughly equivalent to 0.2 cpu seconds on an IBM 370/158) and may fluctuate with the number of users sharing the machine, it still provides a rough basis of comparison for computational results. For instance, we observe that, on average, the running time of Q^2 is roughly twice that of Q^1. Since the Q^2 running time includes the solution of the initial problem $\bar{Q}^2(\emptyset,\emptyset)$, which is equivalent to Q^1, we see that the computational effort required to solve the initial linear program is approximately the same as that for the evaluation of the remaining $\bar{Q}^2(I_0,I_1)$ problems in the enumeration tree.

The second measure in Table 6 reflects the actual number of $\bar{Q}^2(I_0,I_1)$ problems [other than the initial problem $\bar{Q}^2(\emptyset,\emptyset)$] that were solved in the branch-and-bound procedure used to evaluate the enumeration tree. Even though this number is as large as 198 for one test problem (with 14 available work stations and 20 tasks), we observe that, on average, the enumeration tree was evaluated fairly effectively. We attribute this fact to the tight upper bounds produced by the initial problem $\bar{Q}^2(\emptyset,\emptyset)$.

5. MODEL REFINEMENT

The reported work on the station selection and task assignment problem in the context of assembly system design is part of an ongoing research effort concerning the development of adaptable/programmable assembly systems; this work is being done in conjunction with a research team at the Charles Stark Draper Laboratory in Cambridge, MA. Our future work on the assembly system design problem will include the refinement of the current model, given by P, to capture better the essence of the work station selection and task assignment problem. In addition, we intend to explore other aspects of the system design process, as mentioned in Section 1, and to try to understand better the remarkable performance of our approximate solution procedure, as reported in the previous section.

The desire to refine the current formulation is a consequence of our experience using this model, and an earlier version of it (Graves and Whitney [1979]), on industrial studies in which the research team attempted to design an assembly system for an actual product or family of products. Whitney, et al. (1979) and (1981) detail some of these case studies. These studies indicate that the current model is generally very useful to the design engineer. In particular, the model helps the assembly system designer to consider both the economic tradeoffs and the feasibility restrictions inherent in the design of the system. Furthermore, the existence of the model as a design and evaluation tool has made the designer more conscious of the economic criterion, and presumably has resulted in better system designs. Despite this success, the case studies have also identified three restrictions on the current model which limit its effectiveness as a design tool.

First, the current formulation does not accurately model a work station for which an upper limit exists on the number of different tasks which can be assigned to it. This restriction is common for pick-and-place (fixed-stop)

robots which can access only a finite number of locations; in addition this restriction can occur due to limitations on space for fixtures and part feeders, as well as limitations on computer memory. To represent this restriction we need to augment P with the following set of constraints

$$\sum_{j=0}^{N} \sum_{k=j+1}^{N+1} x_{ijk} \leq n_i \qquad \forall i \in I \qquad (12)$$

where n_i is the maximum number of tasks possible at work station i. We can reformulate this augmented problem as a generalized program wherein sets X_i defined in (2) also incorporate (12). This redefinition of X_i results in a more complicated subproblem for the generalized linear programming procedure; otherwise, the solution procedure given in Sections 3 and 4 remains unchanged.

Second, the current formulation does not model economies of scale, in both operation time and cost, that occur when similar tasks are clustered at a single work station. Formulation P assumes that distinct tasks are to be considered independently, except for possible tool change and load times associated with the preceding and succeeding tasks performed at that work station. However, this need not be the case since a particular work station may be capable of performing a cluster of tasks more efficiently than is suggested by the individual task times. To model these economies of scale we need to assume that the task clusters are not too numerous and that these clusters can be identified a priori. Then the dynamic programming procedure used to solve the subproblems S_i may be modified to evaluate each task cluster separately.

Finally, the current formulation does not model the fixed tooling costs associated with the purchase and maintenance of tools necessary for the system design. This cost depends upon the task assignments which dictate the tools needed at each work station. To model the fixed tooling cost in P, we need to

introduce a new set of zero-one variables to denote whether or not a particular tool is needed at a work station. We cannot solve the resulting formulation with the present solution method. However, we note that the inclusion of fixed tooling costs in the model is critical only when these costs are significant and when candidate system designs may require that certain tooling be duplicated at various work stations.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under Grant DAR77-23712 made to The Charles Stark Draper Laboratory, Inc., Cambridge, Massachusetts. The authors wish to acknowledge the very helpful comments and criticisms of Dr. Daniel E. Whitney of The Charles S. Draper Laboratory throughout the duration of this work. The authors are also grateful to Prof. Thomas L. Magnanti of the Massachusetts Institute of Technology and to Mary Anne Causino for their thoughtful reading of the manuscript. Finally, the authors thank the referees for their suggestions which have resulted in an improved paper.

APPENDIX

Tables A1, A2, and A3 contain, respectively, the cost data, the time data, and the tool requirements used in the experimental plan outlined in Section 5.

Table A1: Cost Data for Test Problems

Cost Component	Work Station and Annual Cost ($)							
	G_1	G_2	G_3	S_1	S_2	S_3	S_4	S_5
Fixed Cost[(a)]	72000	46000	70000	15000	15000	15000	15000	15000
Variable Cost[(b)]	500	500	500	100	200	200	200	200

(a) Based on a one-year payback period.

(b) Based on a production rate of one unit per minute for 120,000 minutes per year.

Table A2: Time Data for Test Problems

Time Component	Work Station and Annual Time (1000's of minutes)[a]							
	G_1	G_2	G_3	S_1	S_2	S_3	S_4	S_5
Assembly Time[b]								
Task 1	12	8	10	-	6	-	-	-
Task 2	12	8	10	-	6	-	-	-
Task 3	12	8	10	-	6	-	-	-
Task 4	12	8	10	-	6	-	-	-
Task 5	12	8	10	-	6	-	-	-
Task 6	14	12	12	-	-	6	-	-
Task 7	12	8	10	-	-	6	-	-
Task 8	12	8	10	-	-	6	-	-
Task 9	8	8	10	6	-	-	6	-
Task 10	12	8	10	-	-	6	-	-
Task 11	12	-	10	-	-	-	-	-
Task 12	12	-	10	-	-	-	-	6
Task 13	12	-	10	-	-	-	-	-
Task 14	8	8	10	6	-	-	-	-
Task 15	12	8	10	-	-	-	6	-
Task 16	12	8	10	-	-	-	6	-
Task 17	12	8	10	-	-	-	6	-
Task 18	12	8	10	-	-	-	-	6
Task 19	12	8	10	-	-	-	-	6
Task 20	12	8	10	-	-	-	-	6
Load/Unload Time	2	2	2	2	2	2	2	2
Tool-Change Time[c]	12	8	10	-	-	-	-	-

(a) Based on a production rate of one unit per minute for 120,000 minutes per year.

(b) A dashed (-) entry indicates an infeasible task/work station combination.

(c) A dashed (-) entry indicates that a work station is unable to change tools.

Table A3: Tool Requirements for Test Problems

Number of Task	Tool Required
1	Small three-finger gripper
2	Small three-finger gripper
3	Small three-finger gripper
4	Small three-finger gripper
5	Small three-finger gripper
6	Small three-finger gripper
7	Small three-finger gripper
8	Small three-finger gripper
9	Special screwdriver
10	Small three-finger gripper
11	Contracting collet
12	Rotor-nut tightener
13	Large three-finger gripper
14	Bolt driver[(a)]
15	Special screwdriver
16	Special screwdriver
17	Special screwdriver
18	Rotor-nut tightener
19	Rotor-nut tightener
20	Rotor-nut tightener

(a) Work Station S_1 uses the special screwdriver to perform task 14.

REFERENCES

Bylinsky, G., "Those Smart Young Robots on the Production Line", Fortune, December 17, 1979, pp. 90-93+.

Dantzig, G. B., and Ramser, J. T., "The Truck Dispatching Problem", Management Science, Vol. 6, No. 1, 1959, pp. 81-91.

Dantzig, G. B., and Wolfe, P., "The Decomposition Algorithm for Linear Programming", Econometrica, Vol. 29, No. 4, October 1961, pp. 767-778.

Donlan, T. G., "Automation Moves On: Jobs Are Opening Up on the Assembly Line for Robots", Barrons, June 2, 1980, pp. 4-5+.

Friedrich, O., et al., "The Robot Revolution", Time, December 8, 1980, pp. 72-78+.

Garfinkel, R. S., and Nemhauser, G. L., Integer Programming, New York: John Wiley and Sons, Inc., 1972.

Geoffrion, A. M., "Lagrangean Relaxation for Integer Programming", Mathematical Programming Study 2, North-Holland Pub. Co., 1974, pp. 82-114.

Graves, S. C., and Whitney, D. E., "A Mathematical Programming Procedure for Equipment Selection and System Evaluation in Programmable Assembly", Proceedings of the 18th IEEE Conference on Decisions and Control, Ft. Lauderdale, FL, December 1979, pp. 531-536.

Handler, G. Y., and Zang, I., "A Dual Algorithm for the Constrained Shortest Path Problem", Tel-Aviv University, August 1978.

Ignall, E. J., "A Review of Assembly Line Balancing", Journal of Industrial Engineering, Vol. 16, No. 4, 1965, pp. 244-254.

Lamar, B. W., "Optimal Machine Selection and Task Assignment in an Assembly System Design Problem", M. S. Thesis, Operations Research Center, Massachusetts Institute of Technology, 1980.

Lawler, E. L., Combinatorial Optimization: Networks and Matroids, New York: Holt, Rinehart and Winston, 1976.

Magnanti, T. L., Shapiro, J. F., and Wagner, M. H., "Generalized Linear Programming Solves the Dual", Management Science, Vol. 22, No. 11, July 1976, pp. 1195-1203.

Nevins, J. L., and Whitney, D. E., "Computer-Controlled Assembly", Scientific American, Vol. 238, No. 2, February 1978, pp. 62-74.

"New Hand in the Workplace - The Robot", U.S. News and World Report, December 31, 1979, pp. 73-74.

Nicholson, T., et al., "Blue-Collar Robots", Newsweek, April 23, 1979, pp. 80-81.

"Robots Join the Labor Force", Business Week, June 9, 1980, pp. 62-65+.

Schefter, J., "New Workers on the Assembly Line: Robots that Think", Popular Science, June 1980, pp. 46-57.

Shapiro, J. F., "A Survey of Lagrangean Techniques for Discrete Optimization", in P. L. Hammer, E. L. Johnson, and B. H. Kortz (editors), Annals of Discrete Mathematics 5: Discrete Optimization, North Holland, 1979 pp. 113-138.

Whitney, D. E., et al., "Design and Control of Adaptable-Programmable Assembly Systems - First Report", The Charles Stark Draper Laboratory, Inc., Report No. R-1284, August 1979.

Whitney, D. E., et al., "Design and Control of Adaptable-Programmable Assembly Systems - Final Report", The Charles Stark Draper Laboratory, Inc., Report No. R-1406, January 1981.